(12) UK Patent Application (19) GB (11) 2 362 793 (13) A

(43) Date of A Publication 28.11.2001

(21) Application No 0012684.7

(22) Date of Filing 24.05.2000

(51) INT CL[7]
G06T 17/00

(52) UK CL (Edition S)
H4T TBBA TBEC

(56) Documents Cited
None

(58) Field of Search
UK CL (Edition S) H4T TBBA TBBD TBEC
INT CL[7] G06T 15/00 15/20 15/40 17/00 17/30 17/40
ONLINE: Internet, Epoque.

(71) Applicant(s)
Canon Kabushiki Kaisha
(Incorporated in Japan)
30-2 3-chome, Shimomaruko, Ohta-ku, Tokyo, Japan

(72) Inventor(s)
Jane Haslam
Alexander Ralph Lyons
Richard Ian Taylor

(74) Agent and/or Address for Service
Beresford & Co
2-5 Warwick Court, High Holborn, LONDON,
WC1R 5DH, United Kingdom

(54) Abstract Title
Image processing apparatus

(57) In a 3D computer modelling apparatus 20 input data comprising images of an object recorded at different positions and orientations, data defining the positions and orientations at which the images were recorded, and an initial computer model of the object comprising points in a 3D space, is processed to generate a 3D surface model of the object comprising a polygon mesh together with texture data for each polygon comprising data from an input image. Processing is carried out to generate visibility data which associates at least one input image with each of a number of the 3D points in the initial computer model or with each of a number of voxels in the 3D space in which the initial computer model is defined. The texture data for each polygon is selected in dependence upon the position of the polygon and the stored visibility data.

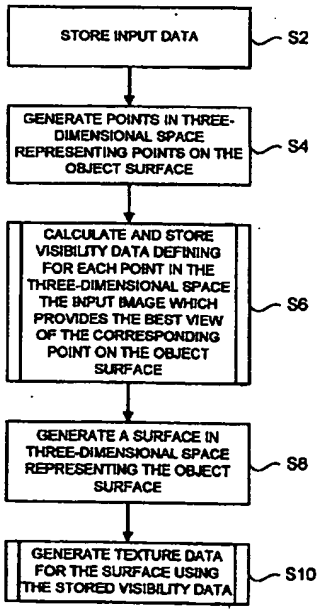

FIG. 3

GB 2 362 793 A

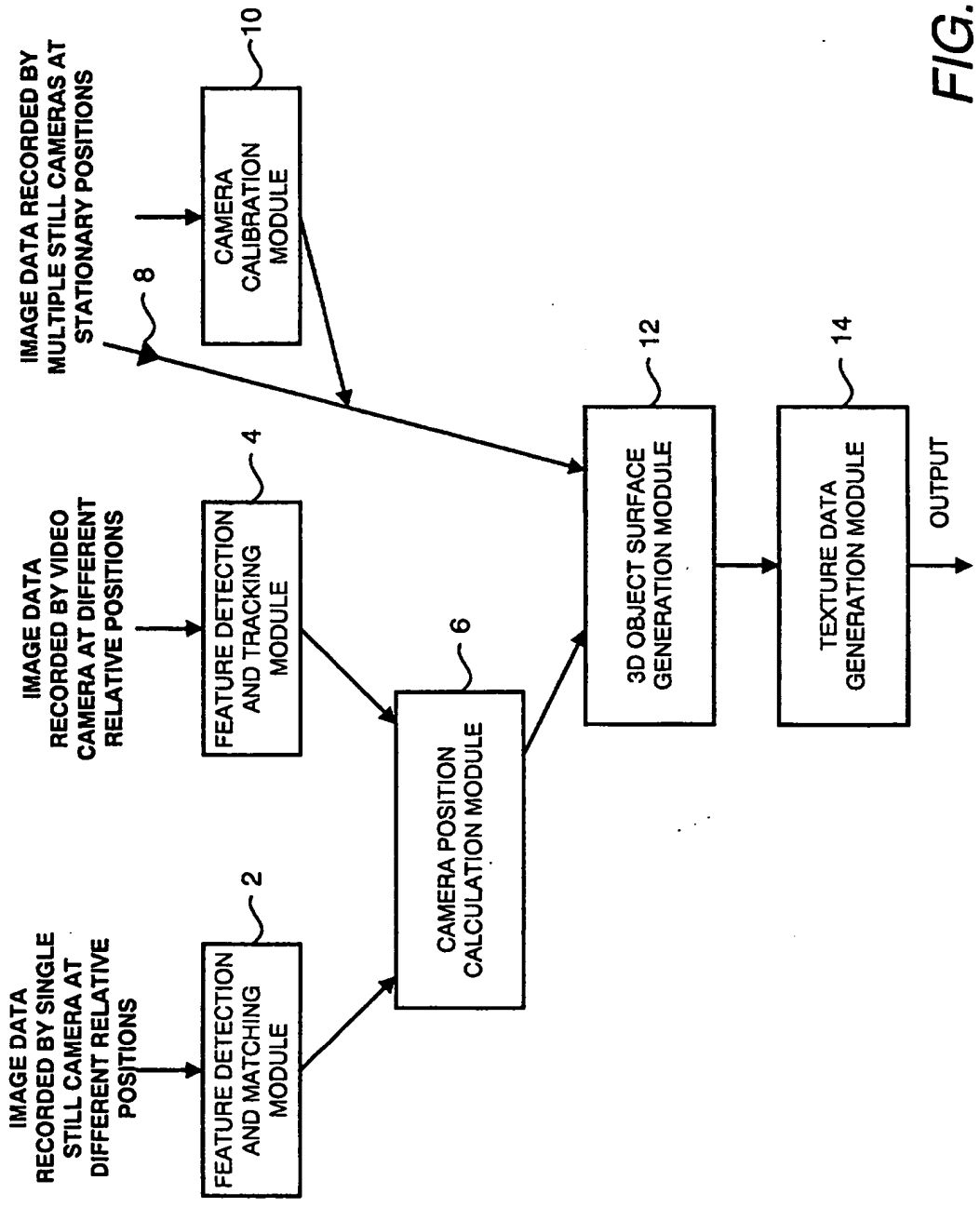
IMAGE DATA RECORDED BY SINGLE STILL CAMERA AT DIFFERENT RELATIVE POSITIONS
FEATURE DETECTION AND MATCHING MODULE
2
IMAGE DATA RECORDED BY VIDEO CAMERA AT DIFFERENT RELATIVE POSITIONS
FEATURE DETECTION AND TRACKING MODULE
4
IMAGE DATA RECORDED BY MULTIPLE STILL CAMERAS AT STATIONARY POSITIONS
8
CAMERA CALIBRATION MODULE
10
CAMERA POSITION CALCULATION MODULE
6
3D OBJECT SURFACE GENERATION MODULE
12
TEXTURE DATA GENERATION MODULE
14
OUTPUT

FIG. 1

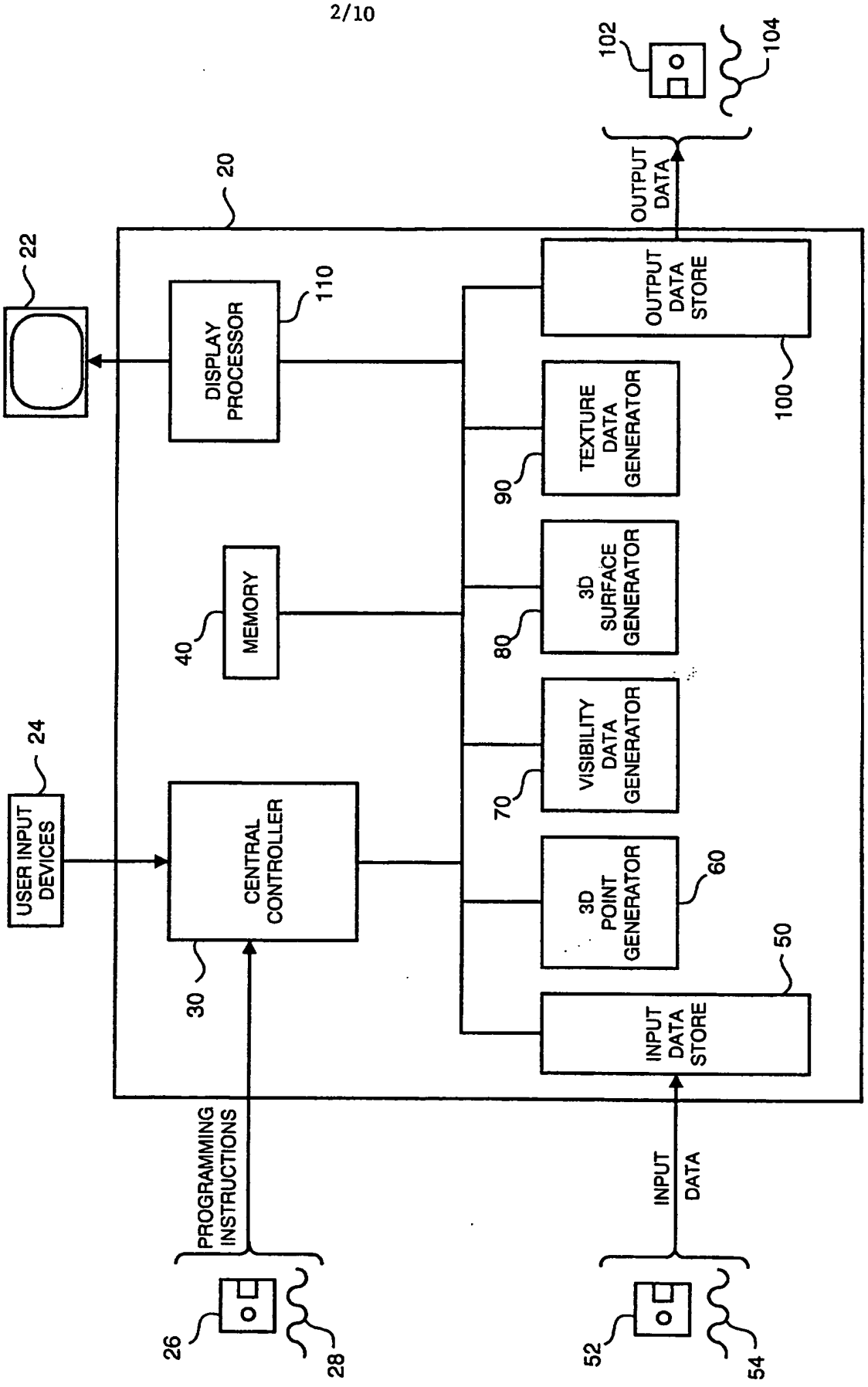
102
104
OUTPUT DATA
20
22
DISPLAY PROCESSOR
110
OUTPUT DATA STORE
100
TEXTURE DATA GENERATOR
90
40
MEMORY
3D SURFACE GENERATOR
80
24
USER INPUT DEVICES
CENTRAL CONTROLLER
VISIBILITY DATA GENERATOR
70
3D POINT GENERATOR
60
30
INPUT DATA STORE
50
PROGRAMMING INSTRUCTIONS
INPUT DATA
26
28
52
54

FIG. 2

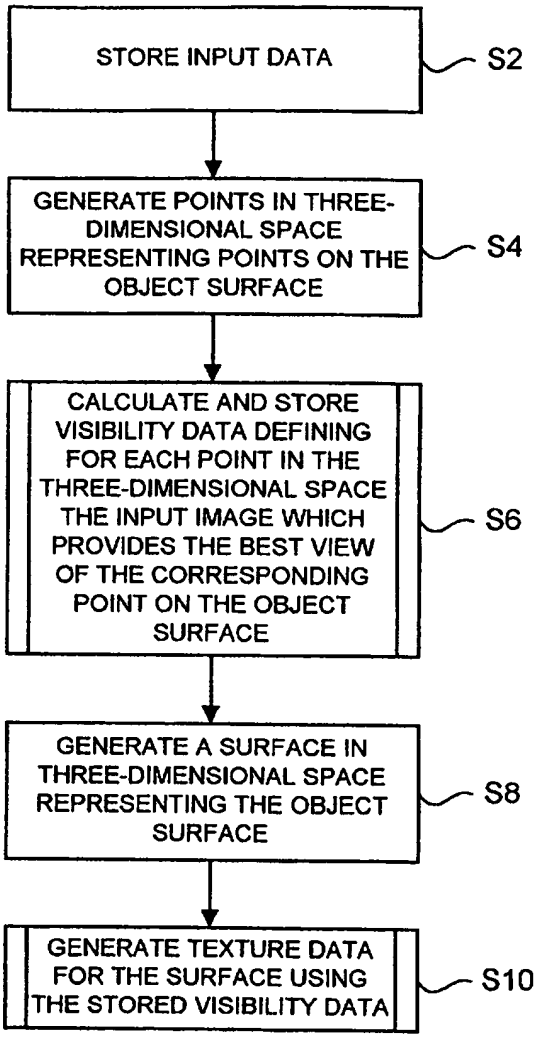
STORE INPUT DATA
S2
GENERATE POINTS IN THREE-DIMENSIONAL SPACE REPRESENTING POINTS ON THE OBJECT SURFACE
S4
CALCULATE AND STORE VISIBILITY DATA DEFINING FOR EACH POINT IN THE THREE-DIMENSIONAL SPACE THE INPUT IMAGE WHICH PROVIDES THE BEST VIEW OF THE CORRESPONDING POINT ON THE OBJECT SURFACE
S6
GENERATE A SURFACE IN THREE-DIMENSIONAL SPACE REPRESENTING THE OBJECT SURFACE
S8
GENERATE TEXTURE DATA FOR THE SURFACE USING THE STORED VISIBILITY DATA
S10

FIG. 3

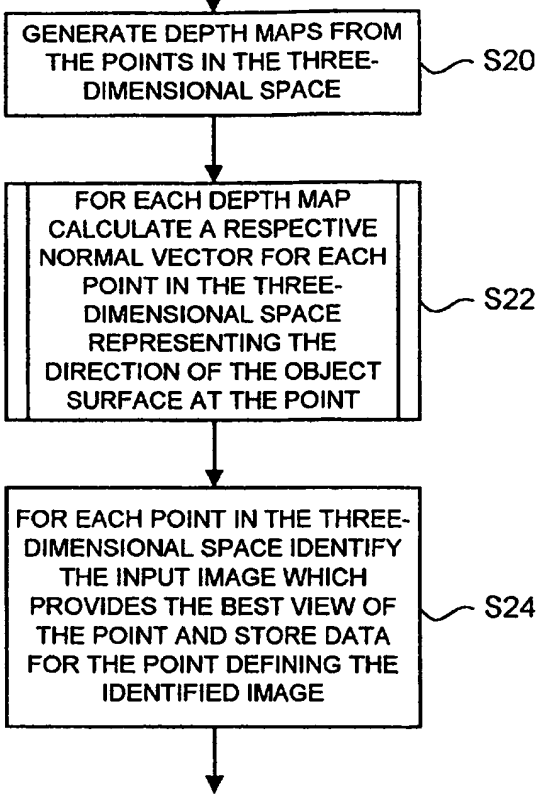
GENERATE DEPTH MAPS FROM THE POINTS IN THE THREE-DIMENSIONAL SPACE
S20
FOR EACH DEPTH MAP CALCULATE A RESPECTIVE NORMAL VECTOR FOR EACH POINT IN THE THREE-DIMENSIONAL SPACE REPRESENTING THE DIRECTION OF THE OBJECT SURFACE AT THE POINT
S22
FOR EACH POINT IN THE THREE-DIMENSIONAL SPACE IDENTIFY THE INPUT IMAGE WHICH PROVIDES THE BEST VIEW OF THE POINT AND STORE DATA FOR THE POINT DEFINING THE IDENTIFIED IMAGE
S24

FIG. 4

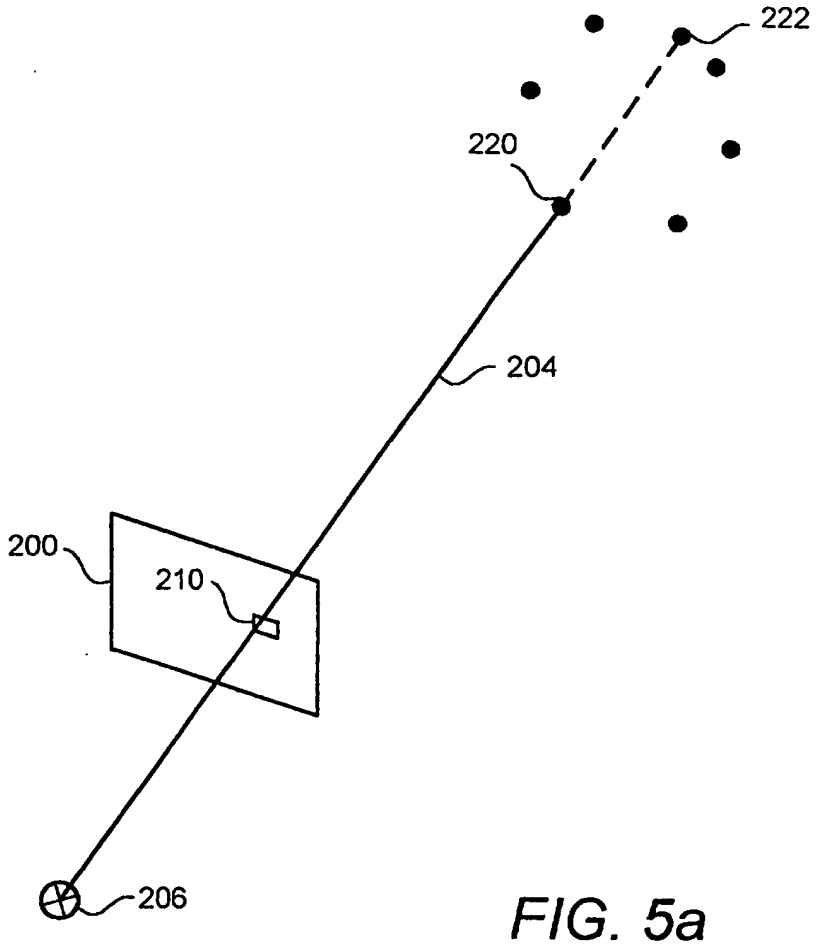

FIG. 5a

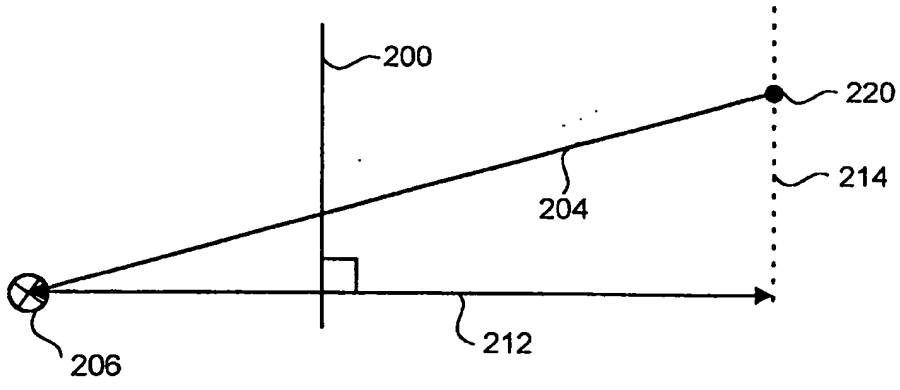

FIG. 5b

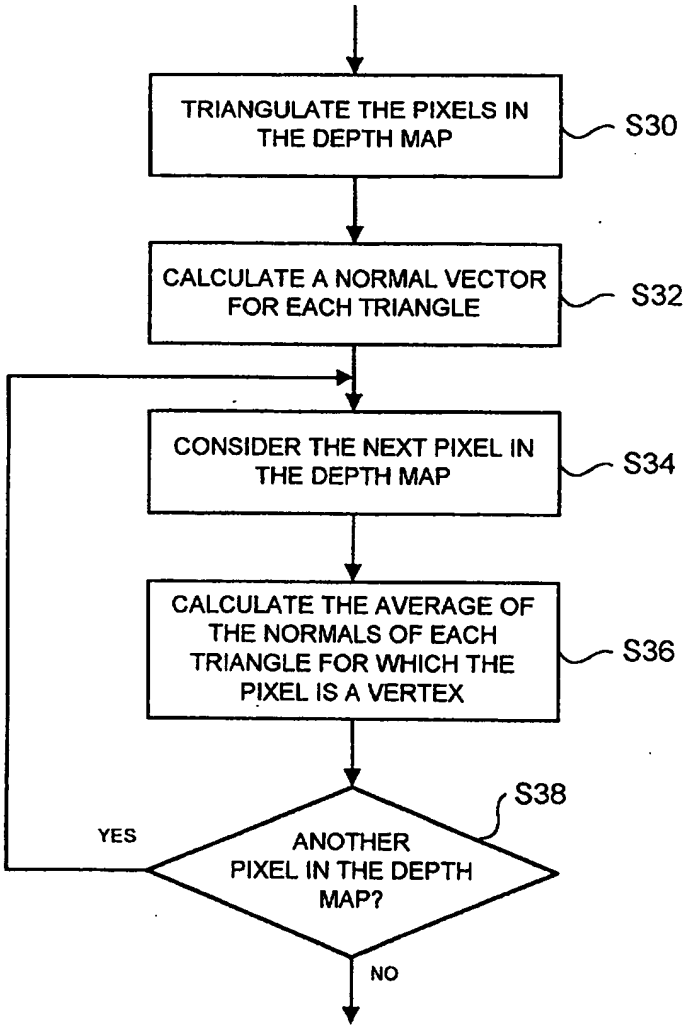
TRIANGULATE THE PIXELS IN THE DEPTH MAP
S30
CALCULATE A NORMAL VECTOR FOR EACH TRIANGLE
S32
CONSIDER THE NEXT PIXEL IN THE DEPTH MAP
S34
CALCULATE THE AVERAGE OF THE NORMALS OF EACH TRIANGLE FOR WHICH THE PIXEL IS A VERTEX
S36
S38
ANOTHER PIXEL IN THE DEPTH MAP?
YES
NO

FIG. 6

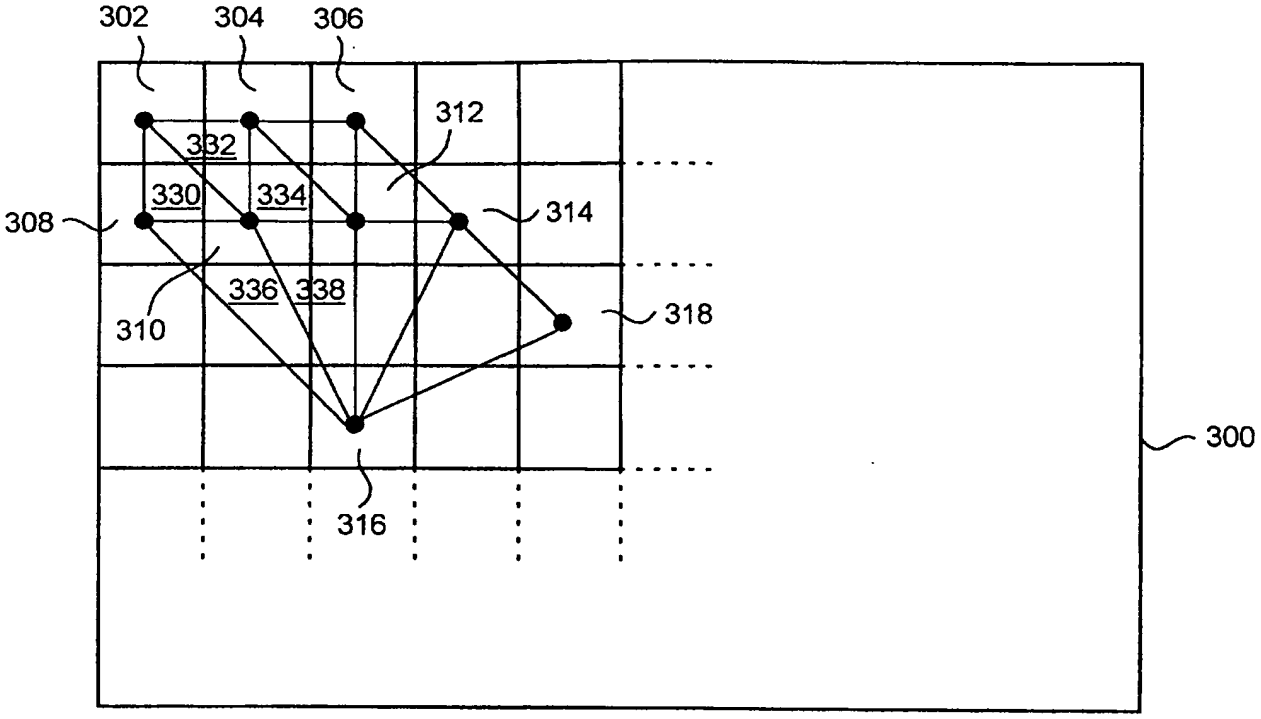

FIG. 7

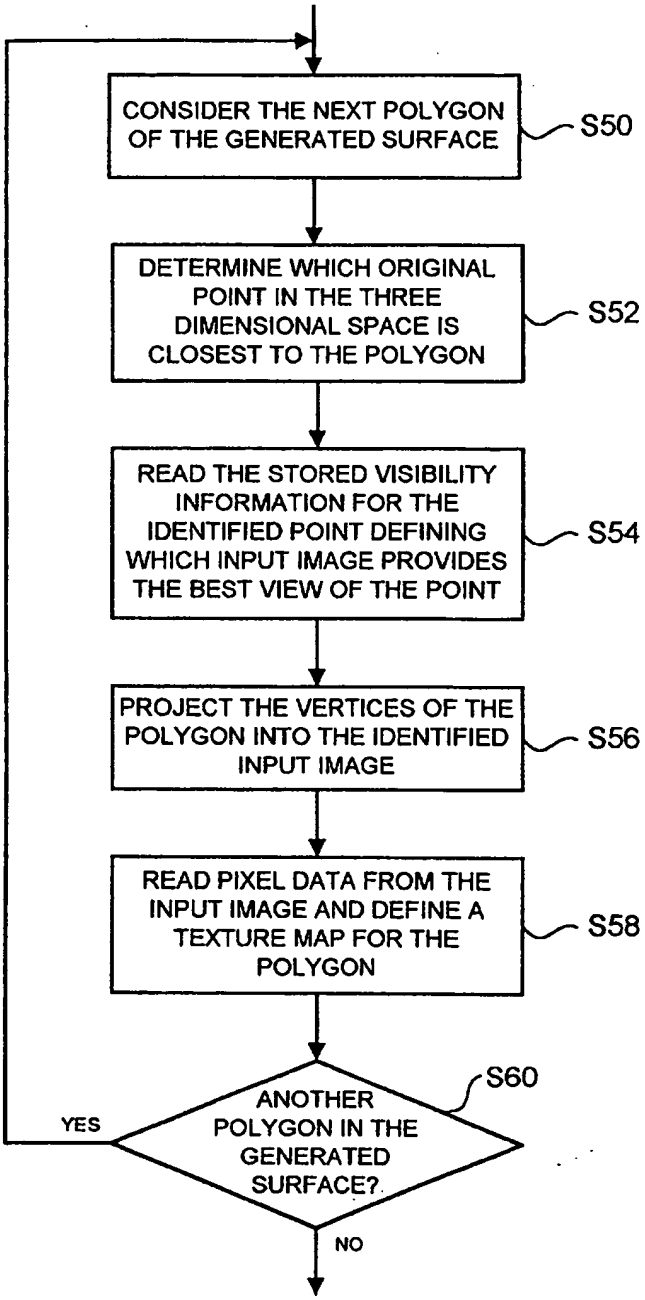
CONSIDER THE NEXT POLYGON OF THE GENERATED SURFACE
S50
DETERMINE WHICH ORIGINAL POINT IN THE THREE DIMENSIONAL SPACE IS CLOSEST TO THE POLYGON
S52
READ THE STORED VISIBILITY INFORMATION FOR THE IDENTIFIED POINT DEFINING WHICH INPUT IMAGE PROVIDES THE BEST VIEW OF THE POINT
S54
PROJECT THE VERTICES OF THE POLYGON INTO THE IDENTIFIED INPUT IMAGE
S56
READ PIXEL DATA FROM THE INPUT IMAGE AND DEFINE A TEXTURE MAP FOR THE POLYGON
S58
ANOTHER POLYGON IN THE GENERATED SURFACE?
S60
YES
NO

FIG. 8

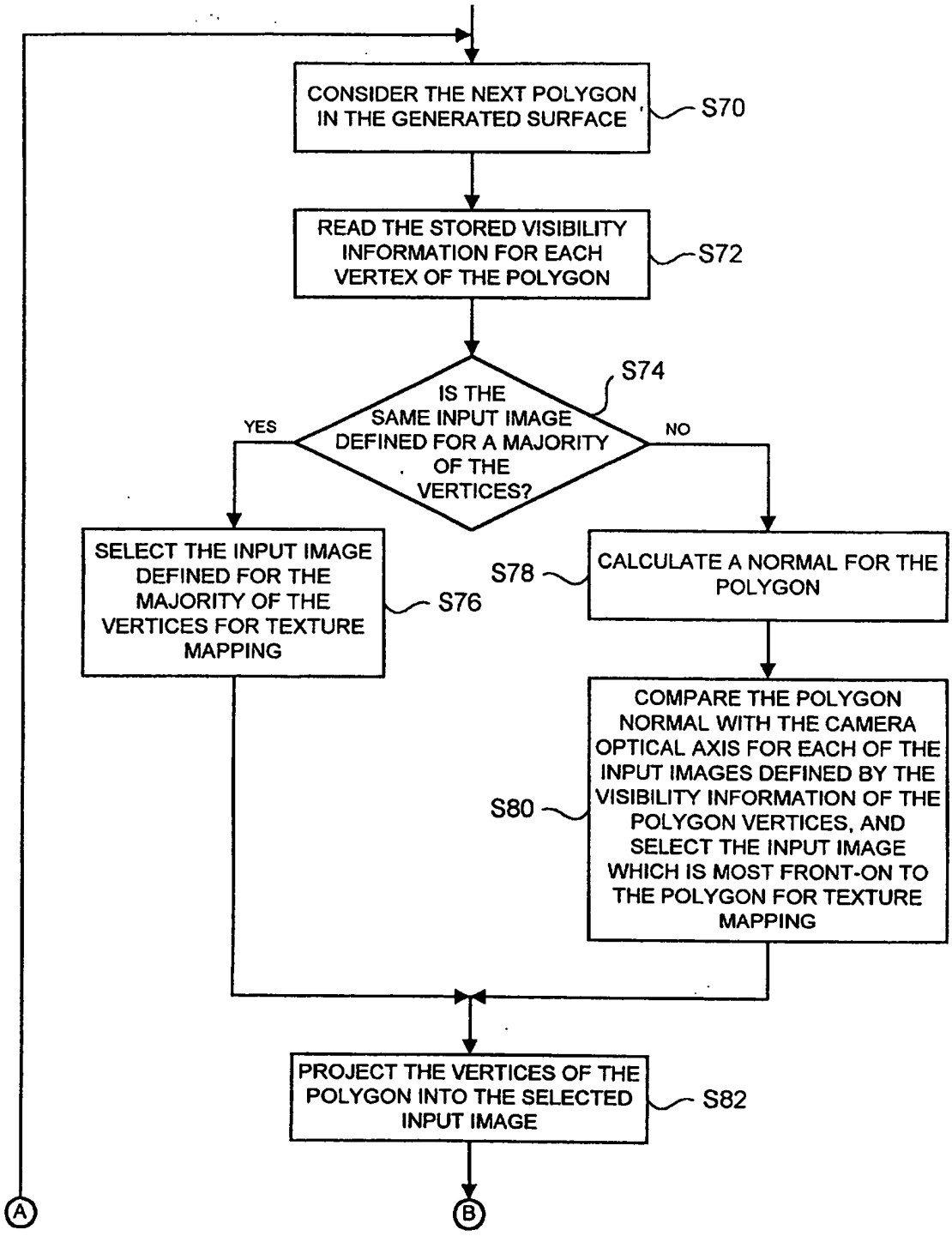
CONSIDER THE NEXT POLYGON IN THE GENERATED SURFACE
S70
READ THE STORED VISIBILITY INFORMATION FOR EACH VERTEX OF THE POLYGON
S72
S74
IS THE SAME INPUT IMAGE DEFINED FOR A MAJORITY OF THE VERTICES?
YES
NO
SELECT THE INPUT IMAGE DEFINED FOR THE MAJORITY OF THE VERTICES FOR TEXTURE MAPPING
S76
S78
CALCULATE A NORMAL FOR THE POLYGON
COMPARE THE POLYGON NORMAL WITH THE CAMERA OPTICAL AXIS FOR EACH OF THE INPUT IMAGES DEFINED BY THE VISIBILITY INFORMATION OF THE POLYGON VERTICES, AND SELECT THE INPUT IMAGE WHICH IS MOST FRONT-ON TO THE POLYGON FOR TEXTURE MAPPING
S80
PROJECT THE VERTICES OF THE POLYGON INTO THE SELECTED INPUT IMAGE
S82
A
B

FIG. 9

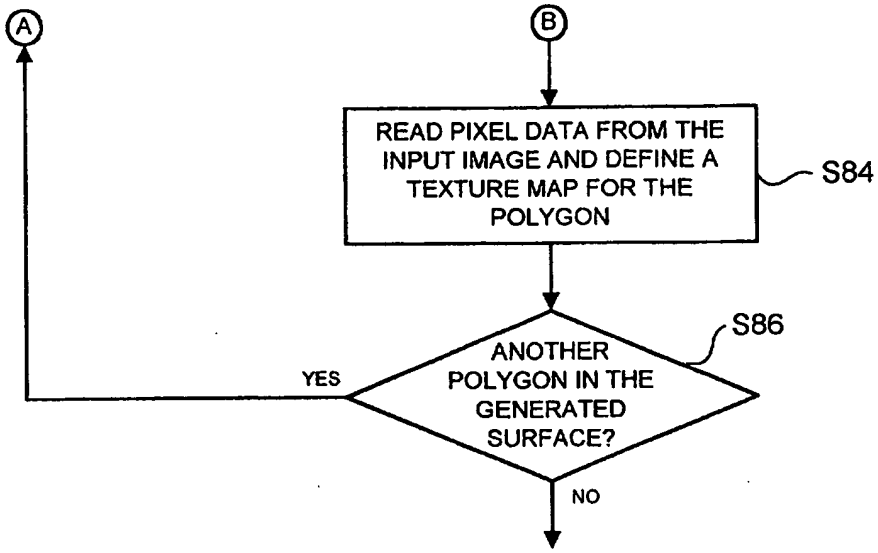
A
B
READ PIXEL DATA FROM THE INPUT IMAGE AND DEFINE A TEXTURE MAP FOR THE POLYGON
S84
S86
ANOTHER POLYGON IN THE GENERATED SURFACE?
YES
NO

FIG. 9 (cont)

IMAGE PROCESSING APPARATUS

The present invention relates to the field of image processing, and more particularly to the processing of data defining a plurality of input images of an object and the positions at which the images were recorded, and data defining a 3D computer model of the object comprising discrete points in a three-dimensional space, to generate data defining a three-dimensional computer surface model representing the surface of the object and texture data for the surface model.

A number of techniques are known for generating three-dimensional computer surface models of an object starting from a plurality of points in a three-dimensional space representing points on the object surface (these 3D points either being defined explicitly in terms of the coordinates in the three-dimensional space, or implicitly as depth maps with data defining the relative positions and orientations of the depth maps and matching points in the depth maps).

These known surface-generation techniques are generally one of two types.

In the first type of surface-generation technique, the discrete points in the three-dimensional space are

connected together with straight lines to create a surface comprising a mesh of triangles, with the 3D points forming the vertices of the triangles. A popular technique for connecting the points to form such a surface is Delaunay triangulation, for example as described in "Three-Dimensional Computer Vision" by Faugeras, MIT Press, ISBN 0-262-06158-9.

In the second type of surface-generation technique, a mesh of polygons (typically triangles) is again generated, but instead of connecting the 3D points so that each vertex of a polygon in the mesh is one of the original 3D points, the surface is generated so that it passes between the positions of the 3D points and represents a "best fit" for the 3D points. In this way, the 3D surface approximately interpolates the 3D points, and the 3D points do not necessarily lie on the generated surface. An example of such a technique is described in "A Volumetric Method for Building Complex Models from Range Images" by Curless and Levoy in Proceedings of SIGGRAPH ACM, 1996, pages 303-312.

To generate texture data for the resulting three-dimensional computer surface model (whether generated by the first or second type of method), each polygon in the surface model is considered in turn, and the polygon normal (that is the vector perpendicular to the surface

of the polygon) is compared with a plurality of images of the object recorded at different positions and orientations (and more particularly to the vector defining the optical axis of the camera when each image was recorded) to select the image which is most front-facing to the polygon. Pixel data is then extracted from the identified image to use as texture data for the polygon in the three-dimensional computer surface model.

This method of generating texture data, however, suffers from a number of problems. In particular, incorrect texture data can be generated because the part of the object surface represented by the polygon may not actually be visible in the selected image because other parts of the object may occlude it (that is, as a result of the position and orientation at which the image was recorded and the position of the part of the object surface represented by the polygon, the part is not visible in the image because it is behind another part of the object surface).

To overcome this problem, it is known to perform a ray-tracing method to test each polygon to determine the input images in which it is visible and then to use only these images in which the polygon can be seen as the images from which to select an image to be used for texture mapping.

However, this method, too, suffers from a number of problems. In particular, the testing of each polygon to determine whether it is visible in each image is computationally expensive and time-consuming.

The present invention has been made with the above problems in mind.

According to the present invention, there is provided a 3D computer modelling apparatus or method in which images of an object and a computer model of the object comprising points in a 3D space are processed to generate a 3D surface model of the object with texture data. Processing is carried out to generate visibility data which associates at least one image with each of a number of positions in the 3D space, and to generate a surface model of the object. Texture data for different parts of the surface model is generated in dependence upon the visibility data.

The positions with which the visibility data associates an image may comprise points (such as the 3D points in the initial object model) and/or regions (such as voxels in the 3D space).

By generating visibility data and using it to select texture data, the speed and accuracy with which texture

data can be generated is greatly increased. In particular, texture data is generated taking into account which parts of the object surface are obscured by others and full ray-tracing visibility calculations are avoided.

Embodiments of the invention will now be described, by way of example only, with reference to the accompanying drawings in which:

Figure 1 schematically shows the components of a modular system in which the present invention is embodied;

Figure 2 schematically shows the components of a first embodiment of the invention, together with the notional functional processing units into which the processing apparatus component may become configured when programmed by programming instructions;

Figure 3 shows the processing operations performed on input data by the apparatus shown in Figure 2 in the first embodiment;

Figure 4 shows the processing operations performed in the first embodiment at step S6 in Figure 3;

Figure 5 schematically illustrates the generation of depth maps in the first embodiment at step S20 in

Figure 4;

Figure 6 shows the processing operations performed in the first embodiment at step S22 in Figure 4;

Figure 7 illustrates an example of the triangulation of pixels at step S30 in Figure 6;

Figure 8 shows the processing operations performed in the first embodiment at step S10 in Figure 3; and

Figure 9 shows the processing operations performed in a third embodiment at step S10 in Figure 3.

First Embodiment

The components of a modular system in which the present invention is embodied are schematically shown in Figure 1.

These components can be effected as processor-implemented instructions, hardware or a combination thereof.

Referring to Figure 1, the components are arranged to process data defining images (still or moving) of one or more objects in order to generate data defining a three-dimensional computer model of the object(s).

The input image data may be received in a variety of ways, such as directly from one or more digital cameras, via a storage device such as a disk or CD ROM, by digitisation of photographs using a scanner, or by downloading image data from a database, for example via a datalink such as the Internet, etc.

The generated 3D model data may be used to: display an image of the object(s) from a desired viewing position; control manufacturing equipment to manufacture a model of the object(s), for example by controlling cutting apparatus to cut material to the appropriate dimensions; perform processing to recognise the object(s), for example by comparing it to data stored in a database; carry out processing to measure the object(s), for example by taking absolute measurements to record the size of the object(s), or by comparing the model with models of the object(s) previously generated to determine changes therebetween; carry out processing so as to control a robot to navigate around the object(s); store information in a geographic information system (GIS) or other topographic database; or transmit the object data representing the model to a remote processing device for any such processing, either on a storage device or as a signal (for example, the data may be transmitted in virtual reality modelling language (VRML) format over the Internet, enabling it to be processed by a WWW browser);

etc.

The feature detection and matching module 2 is arranged to receive image data recorded by a still camera from different positions relative to the object(s) (the different positions being achieved by moving the camera and/or the object(s)). The received data is then processed in order to match features within the different images (that is, to identify points in the images which correspond to the same physical point on the object(s)).

The feature detection and tracking module 4 is arranged to receive image data recorded by a video camera as the relative positions of the camera and object(s) are changed (by moving the video camera and/or the object(s)). As in the feature detection and matching module 2, the feature detection and tracking module 4 detects features, such as corners, in the images. However, the feature detection and tracking module 4 then tracks the detected features between frames of image data in order to determine the positions of the features in other images.

The camera position calculation module 6 is arranged to use the features matched across images by the feature detection and matching module 2 or the feature detection and tracking module 4 to calculate the transformation

between the camera positions at which the images were recorded and hence determine the orientation and position of the camera focal plane when each image was recorded.

The feature detection and matching module 2 and the camera position calculation module 6 may be arranged to perform processing in an iterative manner. That is, using camera positions and orientations calculated by the camera position calculation module 6, the feature detection and matching module 2 may detect and match further features in the images using epipolar geometry in a conventional manner, and the further matched features may then be used by the camera position calculation module 6 to recalculate the camera positions and orientations.

If the positions at which the images were recorded are already known, then, as indicated by arrow 8 in Figure 1, the image data need not be processed by the feature detection and matching module 2, the feature detection and tracking module 4, or the camera position calculation module 6. For example, the images may be recorded by mounting a number of cameras on a calibrated rig arranged to hold the cameras in known positions relative to the object(s).

Alternatively, it is possible to determine the positions

of a plurality of cameras relative to the object(s) by adding calibration markers to the object(s) and calculating the positions of the cameras from the positions of the calibration markers in images recorded by the cameras. The calibration markers may comprise patterns of light projected onto the object(s). Camera calibration module 10 is therefore provided to receive image data from a plurality of cameras at fixed positions showing the object(s) together with calibration markers, and to process the data to determine the positions of the cameras. A preferred method of calculating the positions of the cameras (and also internal parameters of each camera, such as the focal length etc) is described in "Calibrating and 3D Modelling with a Multi-Camera System" by Wiles and Davison in 1999 IEEE Workshop on Multi-View Modelling and Analysis of Visual Scenes, ISBN 0769501109.

The 3D object surface generation module 12 is arranged to receive image data showing the object(s) and data defining the positions at which the images were recorded, and to process the data to generate a 3D computer model representing the actual surface(s) of the object(s) comprising a polygon mesh model.

The texture data generation module 14 is arranged to generate texture data for rendering onto the surface model produced by the 3D object surface generation

module 12. The texture data is generated from the input image data showing the object(s).

Techniques that can be used to perform the processing in the feature detection and matching module 2, feature detection and tracking module 4 and camera position calculation module 6 shown in Figure 1 are described in EP-A-0898245 and EP-A-0901105, the full contents of which are incorporated herein by cross-reference, and also Annex A.

The present invention is embodied in particular as part of the 3D object surface generation module 12 and the texture data generation module 14. Accordingly, a description will now be given of these two modules.

To assist understanding, the processing operations performed by the 3D object surface generation module 12 and the texture data generation module 14 in the embodiment will be described with reference to functional units.

Figure 2 shows examples of such functional units and their interconnections within a single processing apparatus 20 which is arranged to perform the processing operations of the 3D object surface generation module 12 and the texture data generation module 14.

In this embodiment, processing apparatus 20 is a conventional processing apparatus, such as a personal computer, containing, in a conventional manner, one or more processors, memory, graphics cards etc together with a display device 22, such as a conventional personal computer monitor, and user input devices 24, such as a keyboard, mouse etc.

The processing apparatus 20 is programmed to operate in accordance with programming instructions input, for example, as data stored on a data storage medium, such as disk 26, and/or as a signal 28 input to the processing apparatus, for example from a remote database, by transmission over a communication network (not shown) such as the Internet or by transmission through the atmosphere, and/or entered by a user via a user input device 24 such as a keyboard.

When programmed by the programming instructions, the processing apparatus 20 effectively becomes configured into a number of functional units for performing the processing operations which will be described below.

As noted above, examples of such functional units and their interconnections are shown in Figure 2. The units and interconnections illustrated in Figure 2 are, however, notional and are shown for illustration purposes

only to assist understanding; they do not necessarily represent the exact units and connections into which the processor, memory etc of the processing apparatus 20 become configured.

Referring to the functional units shown in Figure 2, a central controller 30 processes inputs from the user input devices 24, and also provides control and processing for a number of the other functional units.

Memory 40 is provided for use by central controller 30 and the other functional units.

Input data store 50 stores input data input to the processing apparatus 20 as data stored on a storage device, such as disk 52, or as a signal 54 transmitted to the processing apparatus 20. The input data comprises data defining a plurality of images of one or more objects recorded at different positions and orientations, together with data defining matching features in the images (that is, the positions in the images of features representing the same physical point on the object surface), data defining the positions and orientations at which the images were recorded, and data defining the intrinsic parameters of the camera (or cameras) which recorded the images, that is, aspect ratio, focal length, principle point (the point at which the optical axis

intersects the imaging plane), first order radial distortion coefficient and skew (the angle between the axes on the pixel grid, because the axes may not be exactly orthogonal).

3D point generator 60 processes the input data to define points in a three-dimensional space which represent physical points on the surface of the object shown in the input images.

Visibility data generator 70 generates data for each of the 3D points generated by 3D point generator 60 defining which of the input images provides the best view of the 3D point (this data subsequently being used to select texture data for the area of the 3D computer model in the vicinity of the 3D point).

3D surface generator 80 generates a mesh of polygons in a three-dimensional space to represent the surface of the object shown in the input images.

Texture data generator 90 uses the visibility data generated by visibility data generator 70 to select pixel data from the input images as texture data for the polygons in the surface mesh generated by 3D surface generator 80.

Output data store 100 stores data defining the 3D surface generated by 3D surface generator 80 and the texture data generated by texture data generator 90, which can then be output under control of central controller 30 as output data, for example as data on a storage device, such as disk 102, or as a signal 104.

Display processor 110, under the control of central controller 30, displays images on display device 22 of the generated 3D computer model of the object from user-selected viewing positions and orientations by rendering the surface model generated by 3D surface generator 80 using the texture data generated by texture data generator 90.

Figure 3 shows the processing operations performed by the processing apparatus 20 in this embodiment.

Referring to Figure 3, at step S2, data input to the processing apparatus 20, for example on disk 52 or as a signal 54, is stored in the input data store 50. As noted above, in this embodiment, the input data comprises data defining a plurality of images of an object, together with data defining matching features in the images (that is, the positions in the images of features representing the same physical point on the object surface), data defining the positions and orientations at

which the images were recorded, and data defining the intrinsic parameters of the camera or cameras which recorded the images, that is, aspect ratio, focal length, principle point (the point at which the optical axis intersects the imaging plane), first order radial distortion coefficient and skew angle.

At step S4, 3D point generator 60 generates points in a three-dimensional space representing physical points on the surface of the object shown in the input images stored at step S2. More particularly, in this embodiment, 3D point generator 60 generates the points in the three-dimensional space using the technique described in EP-A-0898245 and EP-A-0901105 with respect to Figures 41 to 48 therein. Other techniques could, however, be used, such as the technique described in "Metric 3D Surface Reconstruction from Uncalibrated Image Sequences" published in "3D Structure from Multiple Images of Large Scale Environments", Proceedings '98 ISBN 3540653104.

At step S6, visibility data generator 70 calculates and stores visibility data for each 3D point generated at step S4 defining the input image which provides the best view of that point (that is, the best view of the corresponding point on the object surface).

Figure 4 shows the processing operations performed by

visibility data generator 70 at step S6.

Referring to Figure 4, at step S20, visibility data generator 70 uses the 3D points previously calculated at step S4 to generate a depth map for each of the input images.

More particularly, referring to Figures 5a and 5b, for a given input image 200, visibility data generator 70 projects each of the 3D points 202 calculated at step S4 into the image 200 by projecting a ray 204 from the position of the focal point 206 of the camera which recorded the input image to the position of the 3D point calculated by 3D point generator 60. The pixel 210 in the input image 200 which is intersected by the projected ray 204 is determined, this pixel representing the projection of the 3D point in the input image 200. A depth is then defined for the pixel 210 as the distance 212 from the focal point 206 of the camera to a plane 214 which is parallel to the plane of the input image 200 and which contains the 3D point.

In some cases, more than one of the 3D points 202 generated by 3D point generator 60 will project to the same pixel in the input image 200. For example, as shown in Figure 5a, both of the 3D points 220 and 222 project to the pixel 210. In such a case, the depth of the pixel

210 is selected as the smallest depth value (that is, the depth of 3D point 220 in Figure 5a) because the nearest 3D point will obscure 3D points which are further away.

As a result of generating a depth map in this way, a depth value has been calculated and stored for some, but not necessarily all, of the pixels in the given input image 200 (the number of pixels for which a depth is calculated and stored will depend upon the number of 3D points 202 generated at step S4 and their positions).

The processing described above is repeated for each input image to form a respective depth map for each input image.

Referring again to Figure 4, at step S22, visibility data generator 70 calculates, for each depth map generated at step S20, a respective normal vector for each 3D point 202 representing the direction of the object surface at that point.

Figure 6 shows the processing operations performed by visibility data generator 70 at step S22 for a single depth map image (the processing being repeated for each depth map image generated at step S20).

Referring to Figure 6, at step S30, visibility data

generator 70 connects each pixel in the depth map image for which a depth was calculated at step S20 (Figure 4) to form triangles.

An example of the result of this processing is illustrated in Figure 7, which shows the pixels in part of a depth map 300, with depth values having being calculated at step S20 (Figure 4) for the pixels 302, 304, 306, 308, 310, 312, 314, 316 and 318, but none of the other illustrated pixels.

In this embodiment, when triangulating a depth map, visibility data generator 70 is arranged to perform processing to compare the depth values of pixels to be connected and not to connect pixels if the difference in the depth values exceeds a predetermined threshold (set to 20 in this embodiment where there are 256 possible depth values). This is because, a difference in depth values greater than this threshold often indicates an occlusion boundary on the object surface (that is, a part of the object surface which could obscure the points lying to either side of the boundary depending upon the direction from which the object is viewed).

Referring again to Figure 6, at step S32, visibility data generator 70 calculates a normal vector for each triangle generated at step S30. More particularly, the plane in

which the triangle lies is calculated from the depth values of the vertices of the triangle, and the vector which is perpendicular to this plane is calculated as the normal vector.

At step S34, visibility data generator 70 considers the next pixel in the depth map for which a depth is defined (this being the first pixel the first time step S34 is performed). This pixel represents a 3D point generated at step S4.

At step S36, visibility data generator 70 reads the value of the normal calculated at step S32 for each triangle for which the pixel currently being considered is a vertex, and calculates the average of the read normals.

Thus, referring to Figure 7 by way of example, the pixel 310 is a vertex for each of the triangles 330, 332, 334, 336 and 338. Accordingly, at step S36, when considering pixel 310, visibility data generator 70 reads the value of the normal calculated for each of triangles 330, 332, 334, 336 and 338 and calculates the average of these normals. This average value is then used as the value of the normal for the 3D point calculated at step S4 which is represented by pixel 310.

At step S38, visibility data generator 70 determines

whether there is another pixel in the depth map for which a depth has been calculated. Steps S34 to S38 are repeated until each of the pixels in the depth map having a depth value have been processed in the manner described above.

As a result of performing the processing described above with respect to Figure 6 for each depth map, "m" normals will have been calculated for each 3D point generated at step S4, where "m" is the number of input images in which the 3D point is visible (the maximum value of "m" therefore corresponding to the total number of input images).

Referring again to Figure 4, at step S24, visibility data generator 70 performs processing to identify for each 3D point generated at step S4 (Figure 3) the input image which provides the best view of the point.

More particularly, to identify which input image provides the best view of a given 3D point, visibility data generator 70 calculates the respective dot product of each normal vector calculated at step S22 for the 3D point with the vector which is normal to the plane of the input image (depth map) which was used to calculate the normal vector for the 3D point (that is, the camera optical axis). That is, visibility data generator 70

calculates $(\underline{n}_{point})_i \cdot (\underline{n}_{camera})_i$, where $(\underline{n}_{point})_i$ is the normal vector of the 3D point calculated from the "i"th input image and $(\underline{n}_{camera})_i$ is the camera normal vector (optical axis) for the "i"th input image, and "i" runs from 1 to "m", where "m" is the number of input images in which the 3D points is visible (as described above).

The dot product which gives the closest value to -1 is determined and the input image which produced this value is selected as the input image which provides the best view of the 3D point. This is because this input image is the image which was recorded with the smallest angle between the optical axis of the camera and the normal to the 3D point (that is, the input image recorded with the camera most "front on" to the 3D point).

Having identified an input image for a 3D point, visibility data generator 70 stores data associating the identified input image with the 3D point for subsequent use, and repeats the processing for each 3D point generated at step S4.

Referring again to Figure 3, at step S8, 3D surface generator 80 generates data defining a three-dimensional surface representing the surface of the object shown in the input images.

More particularly, in this embodiment, 3D surface generator 80 uses the depth maps generated at step S20 (Figure 4) and performs processing to carry out the method described in "Consensus Surfaces for Modelling 3D Objects from Multiple Range Images" by Wheeler et al in Proceedings ICCV 1998, Bombay, pages 917-924. This creates a surface comprising a mesh of triangles which passes between the points in three-dimensions defined by the depth map images and is a "best fit" to these points, although the points do not necessarily lie on the generated surface.

Data generated at step S8 defining the three-dimensional surface is stored in output data store 100.

At step S10, texture data generator 90 generates texture data for the surface generated at step S8 using the visibility data previously generated by visibility data generator 70 at step S6.

Figure 8 shows the processing operations performed by texture data generator 90 at step S10.

Referring to Figure 8, at step S50, texture data generator 90 considers the next polygon (that is, a triangle in this embodiment) of the surface generated at step S8 (this being the first polygon the first time step

S50 is performed).

At step S52, texture data generator 90 determines which of the 3D points previously generated at step S4 is the closest point to the polygon currently being considered. More particularly, in this embodiment, texture data generator 90 determines which of the 3D points is the closest point to the centre of the polygon currently being considered.

At step S54, texture data generator 90 reads the visibility information (previously stored at step S6) for the point identified at step S52 defining which input image provides the best view of the point.

At step S56, texture data generator 90 projects the vertices of the polygon currently being considered into the input image identified at step S54.

At step S58, texture data generator 90 reads pixel data from the input image defined by the positions of the points projected at step S56 and uses the pixel data to define a texture map for the polygon currently being considered in a conventional manner. The texture map is then stored in the output data store 100.

At step S60, texture data generator 90 determines whether

there is another polygon in the surface generated at step S8, and steps S50 to S60 are repeated until each surface polygon has been processed in the manner described above.

A number of modifications can be made to the first embodiment described above.

For example, in the first embodiment above, at step S8, processing is performed to carry out the method described in "Consensus Surfaces for Modelling 3D Objects from Multiple Range Images" by Wheeler et al in Proceedings ICCV 1998, Bombay, pages 917-924. However, other techniques may be used to generate a surface in a three-dimensional space representing the object surface. For example, the technique described in "On Reliable Surface Reconstruction from Multiple Range Images" by Hilton, Technical Report VSSP-TR-5/95, October 1995, Surrey University or the technique described in "Surface Reconstruction from Unorganised Points" by Hoppe, PhD Thesis, University of Washington 1994 may be used.

Second Embodiment

A second embodiment of the invention will now be described.

The second embodiment comprises the same components as

the first embodiment described above, and the processing performed in the second embodiment is the same as that performed in the first embodiment with the exception of the processing at step S8 (Figure 3) and steps S52 and S54 (Figure 8).

The processing performed in the second embodiment at steps S8, S52 and S54 will therefore now be described.

At step S8, in the second embodiment, 3D surface generator 80 again generates a surface in three-dimensional space representing the object surface using the method described in "Consensus Surfaces for Modelling 3D Objects from Multiple Range Images" by Wheeler et al in Proceedings ICCV 1998, Bombay, pages 917-924.

In this method, a three-dimensional space is divided into a number of voxels, and for each voxel, a signed distance value is calculated and stored representing the distance from the centre point of the voxel to the closest point on the object surface (the sign indicating whether the point is inside, outside or on the surface). The signed distance for a voxel is calculated on the basis of the positional relationship of the centre point of the voxel and the points in three-dimensions defined by the depth maps. A three-dimensional surface comprising a mesh of triangles is then fitted through the voxels in dependence

upon the signed distance values.

In the second embodiment, when performing the processing at step S8, 3D surface generator 80 stores visibility data generated by visibility data generator 70 in the signed distance value for each voxel. More particularly, for each voxel, 3D surface generator 80 determines which of the 3D points generated at step S4 is closest to the centre of the voxel, and then adds data to the signed distance value for the voxel defining the input image identified at step S6 for this closest 3D point.

In the second embodiment, when performing steps S52 and S54 (Figure 8), texture data generator 90 determines which voxels are intersected by the polygon currently being considered and reads the signed distance value calculated as part of the processing performed at step S8 for each of the voxels which are intersected by the polygon. Texture data generator 90 then selects the smallest signed distance value and, at step S54, reads the visibility information defined for the smallest signed distance value which identifies the input image which provides the best view of the polygon.

Third Embodiment

A third embodiment of the invention will now be

described.

The third embodiment comprises the same components as the first embodiment described above, and the processing performed in the third embodiment is the same as that performed in the first embodiment with the exception of the processing performed at step S8 and S10 (Figure 3).

The processing performed in the third embodiment at steps S8 and S10 will therefore now be described.

At step S8, in the third embodiment, 3D surface generator 80 performs processing to generate a surface in three-dimensional space representing the object surface by connecting the 3D points generated at step S4 to form a triangular mesh. More particularly, 3D surface generator 80 performs processing to carry out a Delaunay triangulation of the 3D points in a conventional manner, for example as described in "Three-Dimensional Computer Vision" by Faugeras, MIT Press, ISBN 0-262-06158-9.

The processing operations performed by the texture data generator 90 at step S10 in the third embodiment to generate texture data for the surface using the stored visibility data are shown in Figure 9.

Referring to Figure 9, at step S70, texture data

generator 90 considers the next polygon (triangle) in the surface mesh generated by 3D surface generator 80 at step S8.

At step S72, texture data generator 90 reads the visibility information stored for each vertex of the polygon (since each vertex is one of the 3D points generated at step S4).

At step S74, texture data generator 90 determines whether the visibility information read at step S72 defines the same input image for a majority of the vertices. More particularly, in this embodiment, texture data generator 90 determines whether two or all three of the vertices of the triangle currently being considered have visibility information which defines the same input image.

If it is determined at step S74 that the same input image is defined in the stored visibility information for a majority of the vertices, then, at step S76, texture data generator 90 selects the input image defined for the majority of the vertices as the input image to be used for texture mapping.

On the other hand, if it is determined at step S74 that the same input image is not defined for a majority of the vertices (that is, in this embodiment, the stored

visibility information for each of the three vertices defines a different input image), then, at step S78, texture data generator 90 calculates a normal for the polygon (that is, a vector perpendicular to the plane of the polygon) in a conventional manner.

At step S80, texture data generator 90 compares the polygon normal calculated at step S78 with the camera optical axis for each of the input images defined by the visibility information of the polygon vertices, and selects the input image which is most "front-on" to the polygon as the input image to be used for texture mapping. More particularly, texture data generator 90 calculates the dot product of the polygon normal with the vector representing the camera optical axis of each of the input images defined by the visibility information of the polygon vertices, and selects the input image for which the result of the dot product calculation is closest to -1.

The processing performed by texture data generator 90 at steps S82 to S86 is the same as that performed at steps S56 to S60 in the first embodiment described above, and accordingly will not be described again here.

A number of modifications can be made to all of the embodiments described above.

For example, the input data stored at step S2 may already define depth map images (that is, pixel values as in a conventional image, together with a depth value for some, or all of the pixels). Such input depth map images may be produced by a camera in combination with a range finder or a stereo camera, etc. In such a case, points in a three-dimensional space representing physical points on the surface of the object shown in the images are already defined by the depth map images and the data defining the matching features in the depth map images. Accordingly, step S4 (Figure 3) and step S20 (Figure 4) are then unnecessary.

In the embodiments above, at step S22 (Figure 4), a plurality of normal vectors are calculated for each 3D point - that is, for a given 3D point, a respective normal vector is calculated using each depth map image. Then, at step S24, the input image which provides the best view of the point is identified by calculating the dot product of each normal vector with the camera normal vector for the image which was used to calculate the point normal vector. However, instead, at step S22, the calculated normal vectors may be used to compute an average normal vector for the point in the three-dimensional space and, at step S24, the respective dot product of the average normal vector with the camera normal vector for each of the input images may be

calculated to identify the input image with the best view of the 3D point (that is, the input image having the camera normal vector which produced the dot product value closest to -1).

In the embodiments above, processing is performed at step S6 to calculate and store visibility data before processing is performed at step S8 to generate a surface in three-dimensional space representing the object surface. However, instead, the processing to calculate and store visibility data may be carried out after the processing to generate the 3D surface.

In the embodiments above, processing is performed by a computer using processing routines defined by programming instructions. However, some or all, of the processing could be performed using hardware.

ANNEX A

1. CORNER DETECTION

1.1 Summary

This process described below calculates corner points, to sub-pixel accuracy, from a single grey scale or colour image. It does this by first detecting edge boundaries in the image and then choosing corner points to be points where a strong edge changes direction rapidly. The method is based on the facet model of corner detection, described in Haralick and Shapiro[1].

1.2 Algorithm

The algorithm has four stages:

(1) Create grey scale image (if necessary);

(2) Calculate edge strengths and directions;

(3) Calculate edge boundaries;

(4) Calculate corner points.

1.2.1 Create grey scale image

The corner detection method works on grey scale images. For colour images, the colour values are first converted to floating point grey scale values using the formula:

$$grey\_scale = (0.3 \times red)+(0.59 \times green)+(0.11 \times blue) \quad \text{....A-1}$$

This is the standard definition of brightness as defined by NTSC and described in Foley and van Dam[ii].

1.2.2 Calculate edge strengths and directions

The edge strengths and directions are calculated using the 7 by 7 integrated directional derivative gradient operator discussed in section 8.9 of Haralick and Shapiro[i].

The row and column forms of the derivative operator are both applied to each pixel in the grey scale image. The results are combined in the standard way to calculate the edge strength and edge direction at each pixel.

The output of this part of the algorithm is a complete derivative image.

1.2.3 Calculate edge boundaries

The edge boundaries are calculated by using a zero crossing edge detection method based on a set of 5 by 5 kernels describing a bivariate cubic fit to the neighbourhood of each pixel.

The edge boundary detection method places an edge at all pixels which are close to a negatively sloped zero crossing of the second directional derivative taken in the direction of the gradient, where the derivatives are defined using the bivariate cubic fit to the grey level surface. The subpixel location of the zero crossing is also stored along with the pixel location.

The method of edge boundary detection is described in more detail in section 8.8.4 of Haralick and Shapiro[1].

1.2.4 Calculate corner points

The corner points are calculated using a method which uses the edge boundaries calculated in the previous step.

Corners are associated with two conditions:

(1) the occurrence of an edge boundary; and

(2) significant changes in edge direction.

Each of the pixels on the edge boundary is tested for "cornerness" by considering two points equidistant to it along the tangent direction. If the change in the edge direction is greater than a given threshold then the point is labelled as a corner. This step is described in section 8.10.1 of Haralick and Shapiro[1].

Finally the corners are sorted on the product of the edge strength magnitude and the change of edge direction. The top 200 corners which are separated by at least 5 pixels are output.

2. FEATURE TRACKING

2.1 Summary

This process described below tracks feature points (typically corners) across a sequence of grey scale or colour images.

The tracking method uses a constant image velocity Kalman filter to predict the motion of the corners, and a correlation based matcher to make the measurements of corner correspondences.

The method assumes that the motion of corners is smooth enough across the sequence of input images that a constant velocity Kalman filter is useful, and that corner measurements and motion can be modelled by gaussians.

2.2 Algorithm

1) Input corners from an image.

2) Predict forward using Kalman filter.

3) If the position uncertainty of the predicted corner is greater than a threshold, Δ, as measured by the state positional variance, drop the corner from the list of currently tracked corners.

4) Input a new image from the sequence.

5) For each of the currently tracked corners:

a) search a window in the new image for pixels which match the corner;

b) update the corresponding Kalman filter, using any new observations (i.e. matches).

6) Input the corners from the new image as new points to be tracked (first, filtering them to remove any which are too close to existing tracked points).

7) Go back to (2)

2.2.1 Prediction

This uses the following standard Kalman filter equations for prediction, assuming a constant velocity and random uniform gaussian acceleration model for the dynamics:

$$x_{n+1} = \Theta_{n+1,n}\, x_n \qquad \text{....A-2}$$

$$K_{n+1} = \Theta_{n+1,n}\, K_n \Theta^T_{n+1,n} + Q_n \qquad \text{....A-3}$$

where "x" is the 4D state of the system, (defined by the position and velocity vector of the corner), K is the state covariance matrix, Θ is the transition matrix, and Q is the process covariance matrix.

In this model, the transition matrix and process covariance matrix are constant and have the following values:

$$\Theta_{n+1,n} = \begin{pmatrix} I & I \\ 0 & I \end{pmatrix} \quad \text{....A-4}$$

$$Q_n = \begin{pmatrix} 0 & 0 \\ 0 & \sigma_v^2 I \end{pmatrix} \quad \text{....A-5}$$

2.2.2 Searching and matching

This uses the positional uncertainty (given by the top two diagonal elements of the state covariance matrix, K) to define a region in which to search for new measurements (i.e. a range gate).

The range gate is a rectangular region of dimensions:

$$\Delta x = \sqrt{K_{11}}, \quad \Delta y = \sqrt{K_{22}} \quad \text{....A-6}$$

The correlation score between a window around the previously measured corner and each of the pixels in the range gate is calculated.

The two top correlation scores are kept.

If the top correlation score is larger than a threshold, C_0, and the difference between the two top correlation scores is larger than a threshold ΔC, then the pixel with the top correlation score is kept as the latest measurement.

2.2.3 Update

The measurement is used to update the Kalman filter in the standard way:

$$G = KH^T(HKH^T + R)^{-1} \quad \text{....A-7}$$

$$x \rightarrow x + G(\hat{x} - Hx) \quad \text{....A-8}$$

$$K \rightarrow (I - GH)K \quad \text{....A-9}$$

where "G" is the Kalman gain, "H" is the measurement matrix, and "R" is the measurement covariance matrix.

In this implementation, the measurement matrix and measurement covariance matrix are both constant, being given by:

$$H = (I \ 0) \quad \text{....A-10}$$

$$R = \sigma^2 I \quad \text{....A-11}$$

2.2.4 Parameters

The parameters of the algorithm are:

Initial conditions: x_0 and K_0.

Process velocity variance: σ_v^2.

Measurement variance: σ^2.

Position uncertainty threshold for loss of track: Δ.

Covariance threshold: C_0.

Matching ambiguity threshold: ΔC.

For the initial conditions, the position of the first corner measurement and zero velocity are used, with an initial covariance matrix of the form:

$$K_0 = \begin{pmatrix} 0 & 0 \\ 0 & \sigma_0^2 I \end{pmatrix} \qquad \text{....A-12}$$

σ_0^2 is set to $\sigma_0^2 = 200(\text{pixels/frame})^2$.

The algorithm's behaviour over a long sequence is anyway not too dependent on the intial conditions.

The process velocity variance is set to the fixed value of 50 $(\text{pixels/frame})^2$. The process velocity variance would have to be increased above this for a hand-held

sequence. In fact it is straightforward to obtain a reasonable value for the process velocity variance adaptively.

The measurement variance is obtained from the following model:

$$\sigma^2 = (rK + a) \qquad \text{....A-13}$$

where $K = \sqrt{(K_{11}K_{22})}$ is a measure of the positional uncertainty, "r" is a parameter related to the likelihood of obtaining an outlier, and "a" is a parameter related to the measurement uncertainty of inliers. "r" and "a" are set to r=0.1 and a=1.0.

This model takes into account, in a heuristic way, the fact that it is more likely that an outlier will be obtained if the range gate is large.

The measurement variance (in fact the full measurement covariance matrix R) could also be obtained from the behaviour of the auto-correlation in the neighbourhood of the measurement. However this would not take into account the likelihood of obtaining an outlier.

The remaining parameters are set to the values: Δ=400

pixels^2, C_0=0.9 and ΔC=0.001.

References

i R M Haralick and L G Shapiro: "Computer and Robot Vision Volume 1", Addison-Wesley, 1992, ISBN 0-201-10877-1 (v.1), section 8.

ii J Foley, A van Dam, S Feiner and J Hughes: "Computer Graphics: Principles and Practice", Addison-Wesley, ISBN 0-201-12110-7.

CLAIMS

1. A method of processing input data defining (i) a plurality of input images of an object, (ii) the positions and orientations at which the input images were recorded, and (iii) points in a three-dimensional space representing points on the object surface, to generate data defining a three-dimensional computer model of the object surface and texture data therefor, the method comprising:

processing the input data to generate visibility data identifying an input image for each of at least some of the three-dimensional points;

processing the input data to generate a computer model of the object surface comprising a polygon mesh; and

generating texture data for the polygons in the mesh in dependence upon the generated visibility data.

2. A method according to claim 1, wherein the step of generating visibility data for a three-dimensional point comprises processing the input data to generate an estimate of the direction of the object surface at the three-dimensional point, comparing the estimate with the position and orientation of each input image to generate a value representing a quality of the view that each

input image has of the three-dimensional point, and selecting an input image for the three-dimensional point in dependence upon the generated quality values.

3. A method according to claim 1 or claim 2, wherein, in the step of generating texture data, pixel data from an input image is used to generate the texture data for a given polygon, and the input image from which to take the pixel data is selected as the input image identified in the visibility data for the three-dimensional point which is closest to the given polygon.

4. A method according to claim 3, wherein the input image from which to take the pixel data is selected as the input image identified in the visibility data for the three-dimensional point which is closest to the centre of the polygon.

5. A method according to any preceding claim, wherein, in the step of generating the computer model of the object surface, the object surface is generated by connecting at least some of the points in the three-dimensional space to form polygons having the connected points as vertices.

6. A method of processing input data defining (i) a

plurality of input images of an object, (ii) the positions and orientations at which the input images were recorded, and (iii) points in a three-dimensional space representing points on the object surface, to generate data defining a three-dimensional computer model of the object surface and texture data therefor, the method comprising:

processing the input data to generate visibility data identifying an input image for each of a plurality of voxels in a three-dimensional space;

processing the input data to generate a computer model of the object surface comprising a polygon mesh which intersects voxels in the three-dimensional space; and

generating texture data for the polygons in the mesh in dependence upon the generated visibility data.

7. A method according to claim 6, wherein the step of generating visibility data comprises processing the input data to generate visibility data identifying a respective input image for each of at least some of the three-dimensional points defined in the input data, and identifying an input image for each voxel in dependence upon the position of the voxel relative to the three-dimensional points and the input images identified for the three-dimensional points.

8. A method according to claim 6 or claim 7, wherein, in the step of generating texture data, pixel data from an input image is used to generate the texture data for a given polygon, and the input image from which to take the pixel data is selected from the input images identified in the visibility data for each voxel through which the given polygon passes.

9. A method according to any of claims 6 to 8, wherein, in the step of generating the computer model of the object surface, the object surface is generated such that at least some of the points in the three-dimensional space do not lie on the generated surface.

10. A method according to any preceding claim, wherein the step of generating the visibility data is performed after the step of generating the computer model of the object surface.

11. A method according to any preceding claim, wherein the input data defines the input images and the points in the three-dimensional space as a plurality of depth map images and feature point matches.

12. A method according to any preceding claim, further comprising the step of generating the input data defining

the positions and orientations at which the input images were recorded and the points in the three-dimensional space representing points on the object surface by processing the data defining the plurality of input images.

13. A method according to any preceding claim, further comprising the step of generating a signal conveying the generated polygon mesh and the generated texture data.

14. A method according to claim 13, further comprising the step of recording the signal either directly or indirectly.

15. Apparatus for processing input data defining (i) a plurality of input images of an object, (ii) the positions and orientations at which the input images were recorded, and (iii) points in a three-dimensional space representing points on the object surface, to generate data defining a three-dimensional computer model of the object surface and texture data therefor, the apparatus comprising:

means for processing the input data to generate visibility data identifying an input image for each of at least some of the three-dimensional points;

means for processing the input data to generate a

computer model of the object surface comprising a polygon mesh; and

means for generating texture data for the polygons in the mesh in dependence upon the generated visibility data.

16. Apparatus according to claim 15, wherein the means for generating visibility data is arranged to generate the visibility data for a three-dimensional point by processing the input data to generate an estimate of the direction of the object surface at the three-dimensional point, comparing the estimate with the position and orientation of each input image to generate a value representing a quality of the view that each input image has of the three-dimensional point, and selecting an input image for the three-dimensional point in dependence upon the generated quality values.

17. Apparatus according to claim 15 or claim 16, wherein the means for generating texture data is arranged to generate the texture data for a given polygon using pixel data from an input image, and is arranged to select the input image from which to take the pixel data as the input image identified in the visibility data for the three-dimensional point which is closest to the given polygon.

18. Apparatus according to claim 17, wherein the means for generating texture data is arranged to select the input image from which to take the pixel data as the input image identified in the visibility data for the three-dimensional point which is closest to the centre of the polygon.

19. Apparatus according to any of claims 15 to 18, wherein the means for generating the computer model of the object surface is arranged to model the object surface by connecting at least some of the points in the three-dimensional space to form polygons having the connected points as vertices.

20. Apparatus for processing input data defining (i) a plurality of input images of an object, (ii) the positions and orientations at which the input images were recorded, and (iii) points in a three-dimensional space representing points on the object surface, to generate data defining a three-dimensional computer model of the object surface and texture data therefor, the apparatus comprising:

means for processing the input data to generate visibility data identifying an input image for each of a plurality of voxels in a three-dimensional space;

means for processing the input data to generate a

computer model of the object surface comprising a polygon mesh which intersects voxels in the three-dimensional space; and

means for generating texture data for the polygons in the mesh in dependence upon the generated visibility data.

21. Apparatus according to claim 20, wherein the means for generating visibility data is arranged to process the input data to generate visibility data identifying a respective input image for each of at least some of the three-dimensional points defined in the input data, and to identify an input image for each voxel in dependence upon the position of the voxel relative to the three-dimensional points and the input images identified for the three-dimensional points.

22. Apparatus according to claim 20 or claim 21, wherein the means for generating texture data is arranged to generate the texture data for a given polygon using pixel data from an input image, and is arranged to select the input image from which to take the pixel data from the input images identified in the visibility data for each voxel through which the given polygon passes.

23. Apparatus according to any of claims 20 to 22,

wherein the means for generating the computer model of the object surface is arranged to model the object surface such that at least some of the points in the three-dimensional space do not lie on the generated surface.

24. Apparatus according to any of claims 15 to 23, wherein the apparatus is arranged to process input data defining the input images and the points in the three-dimensional space as a plurality of depth map images and feature point matches.

25. Apparatus according to any preceding claim, further comprising means for generating the input data defining the positions and orientations at which the input images were recorded and the points in the three-dimensional space representing points on the object surface by processing the data defining the plurality of input images.

26. A storage device storing instructions for causing a programmable processing apparatus to become operable to perform a method as set out in at least one of claims 1 to 12.

27. A signal conveying instructions for causing a

programmable processing apparatus to become operable to perform a method as set out in at least one of claims 1 to 12.

PATENTS • DESIGNS • COPYRIGHT • TRADE MARKS
The Patent Office
INVESTOR IN PEOPLE

Application No: GB 0012684.7
Claims searched: All
Examiner: R. F. King
Date of search: 26 June 2001

Patents Act 1977
Search Report under Section 17

Databases searched:

UK Patent Office collections, including GB, EP, WO & US patent specifications, in:

UK Cl (Ed.S): H4T[TBBA, TBBD, TBEC]

Int Cl (Ed.7): G06T15/00, 15/20, 15/40, 17/00, 17/30, 17/40.

Other: ONLINE:Internet, Epoque.

Documents considered to be relevant:

Category	Identity of document and relevant passage	Relevant to claims
A	EP0898245 A [Canon] see abstract	1, 15, 20
"	EP0901105 A " "	"

X	Document indicating lack of novelty or inventive step	A	Document indicating technological background and/or state of the art.
Y	Document indicating lack of inventive step if combined with one or more other documents of same category.	P	Document published on or after the declared priority date but before the filing date of this invention.
&	Member of the same patent family	E	Patent document published on or after, but with priority date earlier than, the filing date of this application.